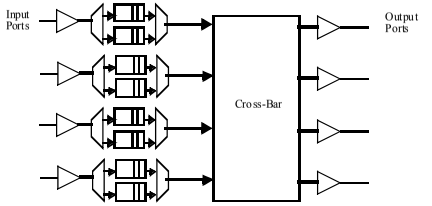
CS 258
Parallel Computer Architecture
Lecture 6

Routing (Con't)

February 11, 2008
Prof John D. Kubiatowicz
http://www.cs.berkeley.edu/~kubitron/cs258

Recall: Deadlock free wormhole networks

- Basic dimension order routing techniques don't work for unidirectional k-ary d-cubes
  - only for k-ary d-arrays (bi-directional)
- Idea: add channels!
  - provide multiple "virtual channels" to break the dependence cycle
  - good for BW too!

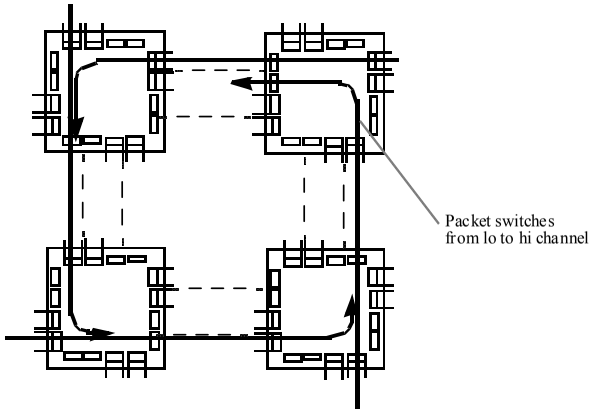


  - Do not need to add links, or xbar, only buffer resources
- This adds nodes to the CDG, remove edges?

Recall: Use of virtual channels for adaptation

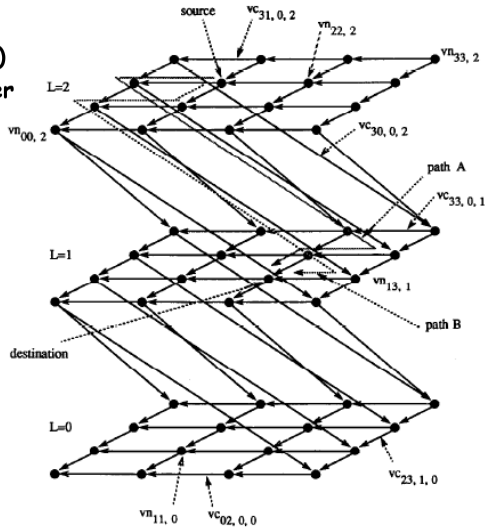
- Want to route around hotspots/faults while avoiding deadlock
- "An adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary n-cubes,"
  - Linder and Harden, 1991
  - General technique for k-ary n-cubes
    - » Requires: 2^{n-1} virtual channels/lane!!!
- Alternative: Planar adaptive routing
  - Chien and Kim, 1995
  - Divide dimensions into "planes",
    - » i.e. in 3-cube, use X-Y and Y-Z
  - Route planes adaptively in order: first X-Y, then Y-Z
    - » Never go back to plane once have left it
    - » Can't leave plane until have routed lowest coordinate
  - Use Linder-Harden technique for series of 2-dim planes
    - » Now, need only 3 × number of planes virtual channels
- Alternative: two phase routing
  - Provide set of virtual channels that can be used arbitrarily for routing
  - When blocked, use unrelated virtual channels for dimension-order (deterministic) routing
  - Never progress from deterministic routing back to adaptive routing

Breaking deadlock with virtual channels



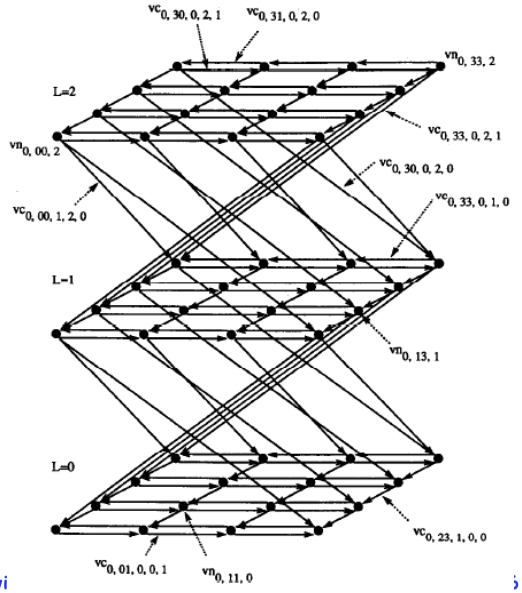
Unidirectional k-ary n-cubes

- n+1 virtual channels
  - (one wrap-around per channel)
  - Switch to new "level" whenever wrap around in any dim
- Any adaptive routing solution is possible as long as:
  - It doesn't use more than n wrap-around channels
  - If want more adaptivity, can add more levels (and more virtual channels)

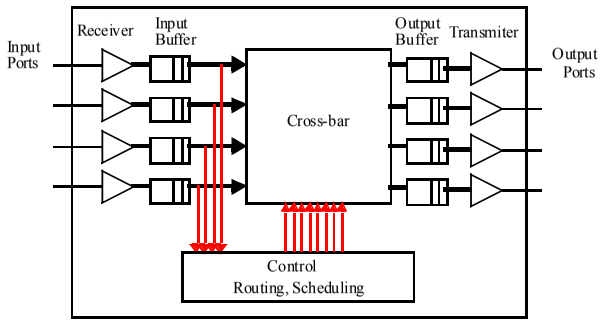


Bidirectional k-ary n-cube

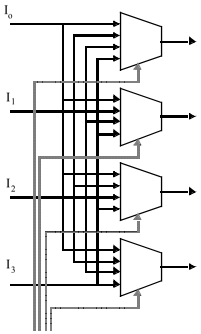
- Need 2^{n-1} virtual networks
  - Except for lowest dimension, only involves single direction

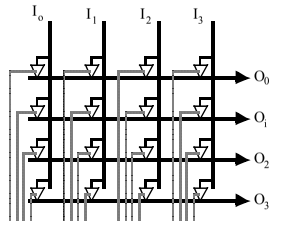


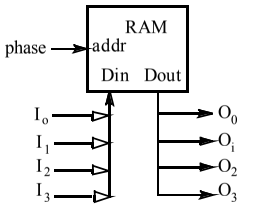
Switch Design



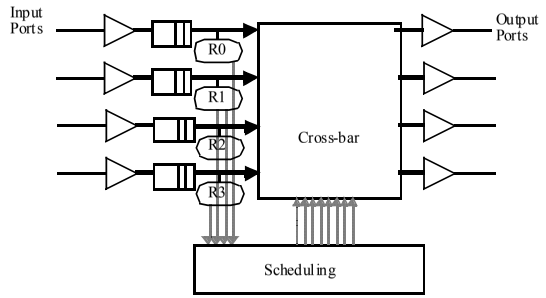
How do you build a crossbar?





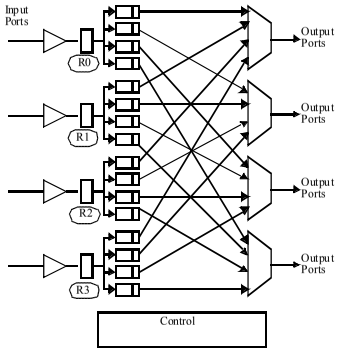


Input buffered swtich



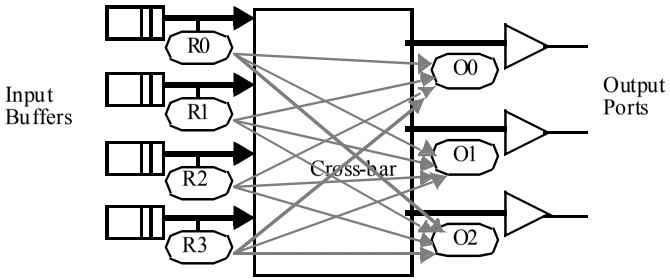
- Independent routing logic per input
  - FSM
- Scheduler logic arbitrates each output
  - priority, FIFO, random
- Head-of-line blocking problem

Output Buffered Switch



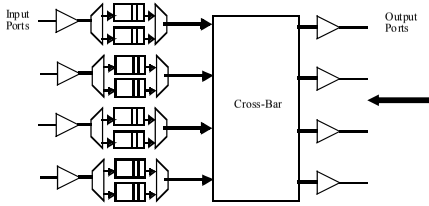
- How would you build a shared pool?

Output scheduling



- n independent arbitration problems?
  - static priority, random, round-robin
- simplifications due to routing algorithm?
- general case is max bipartite matching

When are virtual channels allocated?

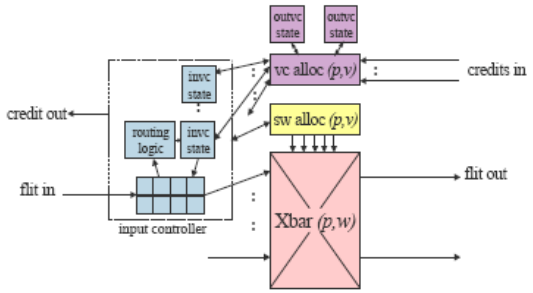


Hardware efficient design
For crossbar

- Two *separate processes:*
  - Virtual channel allocation
  - Switch/connection allocation
- Virtual Channel Allocation
  - Choose route and free output virtual channel
- Switch Allocation
  - For each incoming virtual channel, must negotiate switch on outgoing pin
- In ideal case (not highly loaded), would like to optimistically allocate a virtual channel

Delay analysis of wormhole router

- "A Delay Model and Speculative Architecture for Pipelined Routers"
  - Li-Shiuan Peh and William Dally



- Cannonical model for a virtual-channel-router
  - Separate routing, virtual-channel allocation, and switch allocation

Virtual Channel Analysis

- Identified Various complex modules within router
- Identified a pipelining model
  - Speculative Virtual Channel Allocation
- Developed process-independent models
  - Result permits the evaluation of number of pipelining stages
- How might we evaluate complexity of logic?
  - Ideally, have some measure that reflects algorithmic complexity, not technology-dependent computations
- What is a good normalization?
  - Single, minimum-sized inverter
  - Call the delay of this τ

Process Independent Modeling

- How might we evaluate complexity of logic?
  - Ideally, have some measure that reflects algorithmic complexity, not technology-dependent computations
- What is a good normalization?
  - Single, minimum-sized inverter
  - Call the delay of this τ

Logical Effort: Delay in a Logic Gate

- Express delays in process-independent unit

$$d = \frac{d_{abs}}{\tau}$$

- Delay has two components

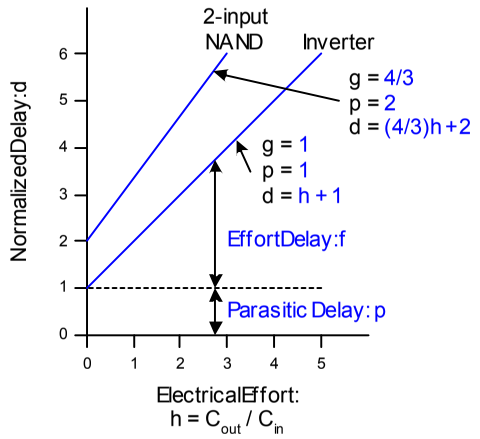
$$d = f + p$$

- Effort delay $f = gh$ (a.k.a. stage effort)
  - Again has two components
- g: *logical effort*
  - Measures relative ability of gate to deliver current
  - $g \equiv 1$ for inverter
- h: *electrical effort* = C_{out} / C_{in}
  - Ratio of output to input capacitance
  - Sometimes called fanout
- p: Parasitic delay
  - Represents delay of gate driving no load
  - Set by internal parasitic capacitance

Delay Plots

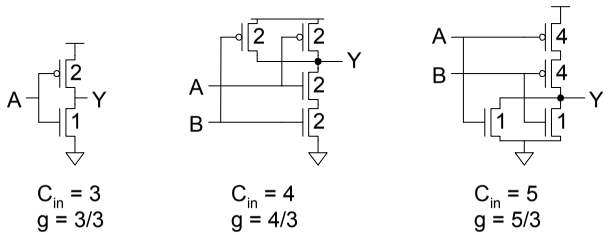
$d = f + p$

$= gh + p$



Computing Logical Effort

- DEF: *Logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current.*
- Measure from delay vs. fanout plots
- Or estimate by counting transistor widths



Catalog of Gates

- Logical effort of common gates

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		4/3	5/3	6/3	(n+2)/3
NOR		5/3	7/3	9/3	(2n+1)/3
Tristate / mux	2	2	2	2	2
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

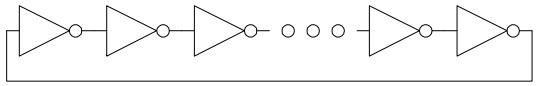
Catalog of Gates

- Parasitic delay of common gates
  - In multiples of p_{inv} ($\approx$1)

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		2	3	4	n
NOR		2	3	4	n
Tristate / mux	2	4	6	8	2n
XOR, XNOR		4	6	8	

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator

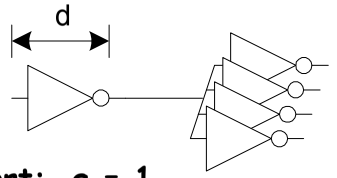


Logical Effort: g = 1
Electrical Effort: h = 1
Parasitic Delay: p = 1
Stage Delay: d = 2
Frequency: f_{osc} = 1/(2*N*d) = 1/4N

31 stage ring oscillator in 0.6 μm process has frequency of ~ 200 MHz

Example: FO4 Inverter

- Estimate the delay of a fanout-of-4 (FO4) inverter



Logical Effort: g = 1
Electrical Effort: h = 4
Parasitic Delay: p = 1
Stage Delay: d = 5

The FO4 delay is about
200 ps in 0.6 μm process
60 ps in a 180 nm process
f/3 ns in an *f* μm process

Multistage Logic Networks

- Logical effort generalizes to multistage networks
- *Path Logical Effort*

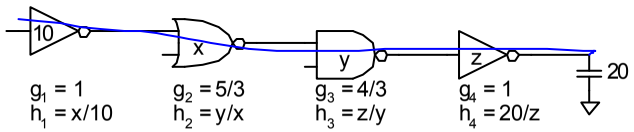
$$G = \prod g_i$$

- *Path Electrical Effort*

$$H = \frac{C_{\text{out-path}}}{C_{\text{in-path}}}$$

- *Path Effort*

$$F = \prod f_i = \prod g_i h_i$$



Multistage Logic Networks

- Logical effort generalizes to multistage networks
- *Path Logical Effort*

$$G = \prod g_i$$

- *Path Electrical Effort*

$$H = \frac{C_{out-path}}{C_{in-path}}$$

- *Path Effort*

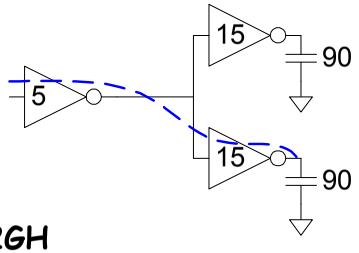
$$F = \prod f_i = \prod g_i h_i$$

- Can we write F = GH?

Paths that Branch

- No! Consider paths that branch:

G = 1
H = 90 / 5 = 18
GH = 18
h_1 = (15 +15) / 5 = 6
h_2 = 90 / 15 = 6
F = $g_1g_2h_1h_2$ = 36 = 2GH



Branching Effort

- Introduce *branching effort*
  - Accounts for branching between stages in path

$$b = \frac{C_{\text{on path}} + C_{\text{off path}}}{C_{\text{on path}}}$$

- Now we compute the path effort Note:
  - F = GBH

$$B = \prod b_i \qquad \prod h_i = BH$$

Multistage Delays

- Path Effort Delay
- Path Parasitic Delay
- Path Delay

$$D_F = \sum f_i$$

$$P = \sum p_i$$

$$D = \sum d_i = D_F + P$$

Designing Fast Circuits

$$D = \sum d_i = D_F + P$$

- Delay is smallest when each stage bears same effort

$$\hat{f} = g_i h_i = F^{\frac{1}{N}}$$

- Thus minimum delay of N stage path is

$$\boxed{D = NF^{\frac{1}{N}} + P}$$

- This is a key result of logical effort
  - Find fastest possible delay
  - Doesn't require calculating gate sizes

Gate Sizes

- How wide should the gates be for least delay?

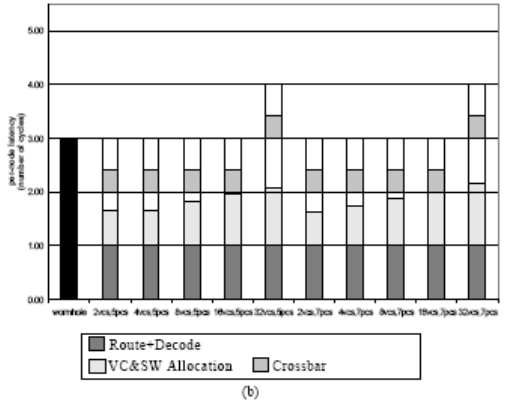
$$\hat{f} = gh = g\frac{C_{out}}{C_{in}}$$

$$\Rightarrow C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$$

- Working backward, apply capacitance transformation to find input capacitance of each gate given load it drives.
- Check work by verifying input cap spec is met.

How does this relate to Router Model?

- Example of results possible:



  - Evaluation of latency as function of VC-allocation algorithm complexity
  - Develop VC-allocator module as circuit, compute logical effort

Summary

- Deadlock-free if channel dependence graph is acyclic
  - limit turns to eliminate dependences
  - add separate channel resources to break dependences
  - combination of topology, algorithm, and switch design
- Switch design issues
  - input/output/pooled buffering, routing logic, selection logic
- Logical Effort
  - Technology-independent delay model: compared with inverter
  - *d = gh + p*
  - *g:logical effort, h:electrical effort, p:parisitic delay*
- "A Delay Model and Speculative Architecture for Pipelined Routers"
  - Speculation on virtual-channel allocation
  - Improves: low conflict latency and throughput